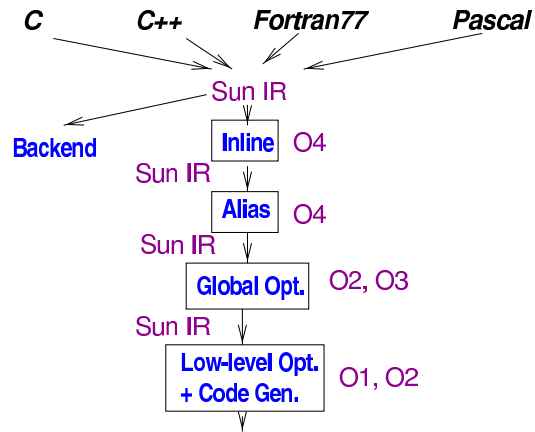
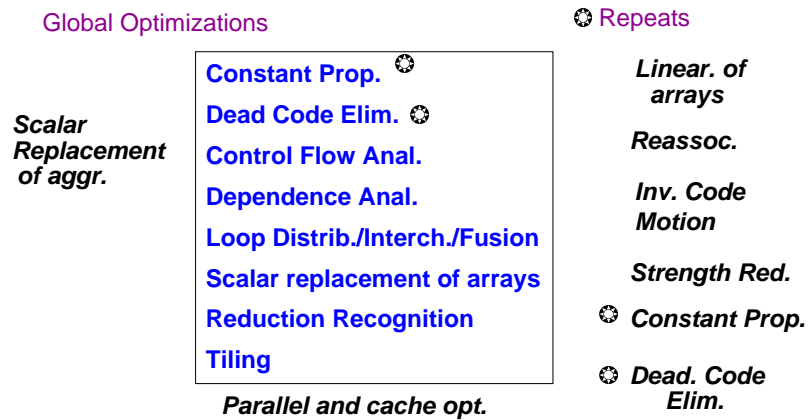


Sun Sparc Compilers



1

Sun Sparc Compilers



2

Sun Sparc Compilers

Low Level Optimizations

⊗ Repeats

Instruction Sel.

Inline

Local Opt.

Dependence Anal.

SW Pipelining

Instruction Sched. ⊗

Register Allocation (graph coloring)

Instruction Sched. ⊗

3

GC Compiler

Many front ends and back ends!

⊗ Repeats

Uses RTL, machine-dependent IL (*from machine description*)

⊗ Jump Opt.

⊗ CSE, CP, branch folding

Loop optimizations, CSE ⊗

DFA: live registers

Instruction combining: peephole opt., instr. sel.

⊗ Instruction Sched.

Local, global Reg. Alloc

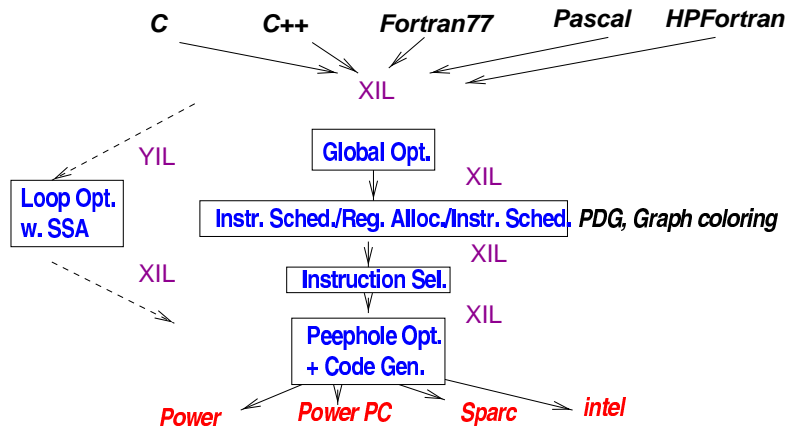
⊗ Instruction Sched.

⊗ Jump Opt.

Branch Opt.

4

IBM XL Compilers



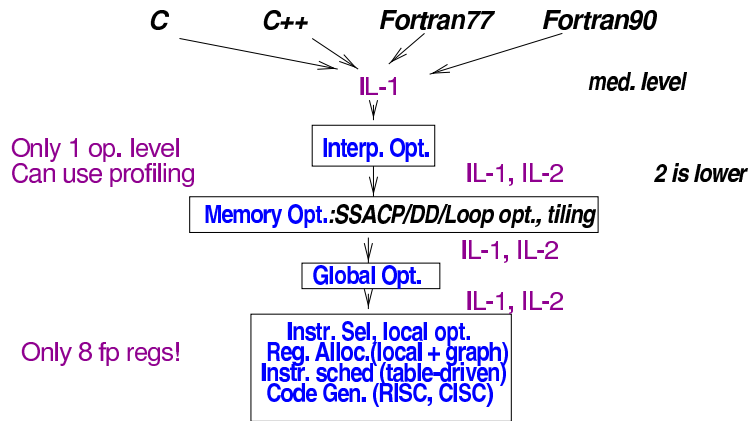
5

IBM XL Compilers

- Global Optimizations
- Branch Opt.
 - Heuristic-based Inline
 - (aggressive) Value No. ⊛ Repeats
 - ⊛ Global CSE, Loop-inv. code motion
 - Store motion and elim.
 - Reassoc.
 - Global Constant Prop.
 - Dead Code Elim.
 - Value No.
 - ⊛ Global CSE
 - Dead Code Elim. ⊛

6

Intel 386 Family Compilers



9

Intel 386 Family Compilers

- Global Optimizations ⊛ Repeats
- Global Constant Prop.
 - Dead Code Elim. ⊛
 - ⊛ Copy Prop.
 - Partial Red. Elim.
 - ⊛ Copy Prop.
 - Dead Code Elim. ⊛

10

Tera Compiler

F77, C, C++, parallel assertions, futures

Whole program compilation

Interprocedural Analysis and Optimization

Call graph, automatic and user-directed inlining, dep info.

Incremental recompilation

Optimizations

Classical scalar

Parallelization (multiple nodes, threads)

Loop unrolling, SW pipelining

Canal (Compiler Analysis Tool)

Reports compiler transformations, # ops, # streams needed

11

IA64 Architecture

Explicit Parallelism: Instruction Level

can execute multiple instructions simultaneously

like VLIW

Large number of registers : 128

Predication

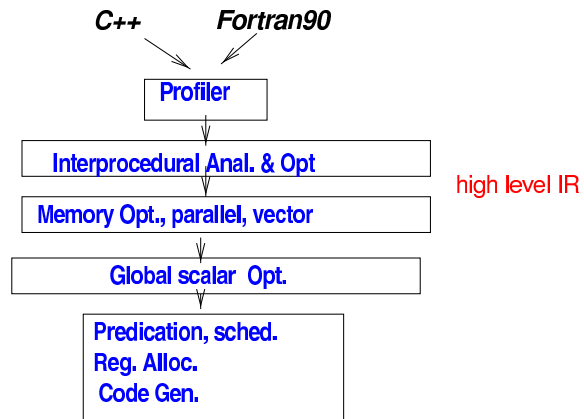
To enhance ILP, help with hard to predict branches

Speculation

To enhance ILP, minimize memory latency

12

Intel IA-64 Compiler



13

Intel IA64 Compiler

Profiler

Static: heuristics to estimate frequencies

Dynamic: Profile feedback

Instrumented compilation, execution on typical input set

Second compilation annotates IR with observed frequencies

Used to guide optimizations

Integrate procedures

Scheduling

14

Memory Optimization

Data dependence and Reuse analysis, CP, copy prop., dead code elim.

Cache:

Loop reversal, interchange, skew, scale, fusion, distribution

Blocking & unrolling (combined)

Register:

Blocking

Different levels of memory:

Data Prefetching

15

Scalar & Later Optimizations

Extended Partial Redundancy Elimination

Including speculation on frequent paths

Removal of redundant loads and stores

Predication

Software Pipelining

Overlap execution of several iterations

Global Code Scheduler

Register Allocation (graph coloring)

Local Code Scheduler

16